

## EAST Search History


Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S1	169	717/137.CCLS.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/04/22 15:04
S2	1	(byron near vargas).in.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/10/23 09:36
S3	460819	(translat\$3 or transform\$5 or conver\$5) with (code or object or instruction or language)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/04/18 09:55
S4	8971	S3 and (oop or (object adj oriented adj programming) or (high-level adj programming) or (high adj level adj programming))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/04/18 09:57
S5	6421	S4 and (database or library)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/04/18 09:58
S6	2990	S5 and analyz\$3	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/04/18 09:59
S7	1430	S6 and pars\$3	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/04/18 09:59
S8	411	S7 and (emulat\$3 or mimic\$4 or imitat\$3)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/04/18 10:01

## EAST Search History

S9	86	S8 and (data adj manipulats)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/04/18 10:11
S10	39	"5768564"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/04/18 10:38
S11	0	("2005/0273315").URPN.	USPAT	OR	ON	2007/04/18 10:39
S12	1	"200211015"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/04/18 11:05
S13	1	pct/fi02/00411.pct.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/04/18 10:57
S14	1	2001FI-20011015	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/04/18 11:05
S15	2	"20050273315"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/04/18 12:39
S16	1	"200293371".PN.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/04/18 13:15
S17	41	"4729096"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/04/18 13:16

## EAST Search History

S18	2	"4729096".PN.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/04/18 14:22
S19	95	S8 and (bidirection\$3 or bi-direction\$3)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/04/18 14:24
S20	12	S8 and ((bidirection\$3 or bi-direction\$3) with (translat\$3 or transform\$5 or conver\$5))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/04/18 14:32
S21	37	S7 and ((bidirection\$3 or bi-direction\$3) with (translat\$3 or transform\$5 or conver\$5))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/04/18 14:37
S22	83	S19 not S21	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/04/18 14:37

<b><i>Index of Claims</i></b> 	<b>Application/Control No.</b> 10716099	<b>Applicant(s)/Patent Under Reexamination</b> VARGAS, BYRON D.
	<b>Examiner</b> Nguyen, Phillip H	<b>Art Unit</b> 2193

✓	<b>Rejected</b>
=	<b>Allowed</b>

-	<b>Cancelled</b>
÷	<b>Restricted</b>

N	<b>Non-Elected</b>
I	<b>Interference</b>

A	<b>Appeal</b>
O	<b>Objected</b>

<input type="checkbox"/> Claims renumbered in the same order as presented by applicant <input type="checkbox"/> CPA <input type="checkbox"/> T.D. <input type="checkbox"/> R.1.47										
CLAIM		DATE								
Final	Original	10/23/2006	04/18/2007							
	37	✓	-							
	38	✓	-							
	39	✓	-							
	40	✓	-							
	41	✓	-							
	42	✓	-							
	43	✓	✓							
	44	✓	✓							
	45	✓	✓							
	46	✓	-							
	47	✓	✓							
	48	✓	✓							

## Object Pascal

There is no official Object Pascal standard, but a technical report was published in 1993 outlining the recommended standards for Object Pascal. A draft of this report is available on the web at <http://pascal-central.com/ooe-stds.html>. As I am not an avid user of Object-Oriented programming, only an outline of the draft report will be included in this paper (Figure 7 below). Further information can be obtained at the above web page.

- I. Introduction
- II. Scope
- III. References
- IV. Definitions
- V. Definitional Conventions
- VI. Compliance
- VII. Object Extensions
  - 1. Class Definition
    - Extension of the Type System, Restrictions on Class Definitions, Contents and Syntax of Class Definitions  
*(Kinds, Inheritance, Fields, Methods, Constructors, Destructors)*
    - Scope of Entities Defined in a Class, Class Definitions
  - 2. Kinds of Classes  
*(Concrete, Abstract, Property, Type Model, Views)*
  - 3. Inheritance  
*(The Root Class, Multiple Inheritance, Name Conflicts, Overriding, Abstract Methods, Constructors, and Destructors)*
  - 4. Syntax
  - 5. Object Access
    - The Object Model, Implicit Parameter Self
    - Polymorphism during Construction and Destruction
    - Implicit References, Field References, Inherited, Reference Type Coercion Operations  
*(Compatibility, Methods Activation, Constructors & Destructors, Assignment, Comparison, Parameter Passing, Membership)*
  - 6. Predefined Entities
    - Null, Copy, Root  
*(Create, Destroy, Clone, Equal)*
    - TextWritable  
*(ReadObj and WriteObj)*
  - 7. Signatures
  - 8. With Statement
  - 9. Procedure, Function, Constructor, and Destructor Declarations
  - 10. Changes to Export Clause
  - 11. Visibility
  - 12. Extended Pascal Features
  - 13. Suggested Changes to Extended Pascal

# Object Oriented Programming in C

Laurent Deniau

March 10, 2001  
revised May 17, 2001

[ [Introduction](#) | [Object Model](#)

[Object](#) | [Class](#) | [Messages](#) | [Methods](#) | [Encapsulation](#) | [Interface](#) | [Implementation](#)  
[Inheritance](#) | [Multiple Inheritance](#) | [Polymorphism](#) | [Abstract Class](#) | [Genericity](#) | [Exceptions](#)  
[Debugging](#) | [ISO C99](#) | [Performances](#) | [Keywords](#) | [Examples](#) | [References](#) | [Mailing List](#) | [ChangeLog](#) ]

my OOPC home page

## Introduction

This paper presents some programming techniques that allow large projects based on ISO C89 to get the benefits of object oriented design. It doesn't intend to be a course on OOP techniques and assumes the reader to have a good knowledge of the C language. Since OOPC is based on the C++ *Object Model*, a good knowledge of C++ may help to understanding it better. These techniques may be useful for programmers who have not a C++ compiler for their architecture (calculators, small systems, embedded systems). It may also be useful for people who are disappointed by C++ compilers which do not behave like the norm says or even do not support all the C++ features or by C++ APIs that change from time to time. In fact, I don't know (at the revised date of this paper) any compiler which fully support the norm C++98. It is clear that the techniques presented here have not the pretension to replace C++, that is impossible without a *cfront* translator (OOPC uses only C macros and few C lines), but it provides enough to do serious OOP:

- The *procedural model* using C functions...
- The *abstract data type model* using public interface and private implementation as well as data and names encapsulation.
- The *object-oriented model* using (multiple) inheritance and polymorphism which allows to manipulate different object types through a common interface.

OOPC also provides mechanisms to handle easily:

- Genericity (C++: templates).
- Exceptions (C++: exceptions, try, catch, throw, auto\_ptr).
- Debugging (dynamic allocation, function call, object construction).

Comparing to the C++ *Object Model*, OOPC does not provide:

- Automatic object construction and destruction (must be explicitly called).
- Automatic array of objects construction and destruction (must be done by user).
- Non-polymorphic objects (all objects are polymorphic).
- Virtual inheritance (too complex and not essential).